REMARKS

This communication is a full and timely response to the aforementioned non-final Office Action dated April 5, 2007. By this communication, claims 1, 4, 5 and 7-10 are amended. Claims 2, 3, 6 and 11-16 are not amended. Thus, claims 1-16 are pending in the application. Reexamination and reconsideration of the application are respectfully requested in view of the foregoing amendments and the following remarks.

I. Amendments to Claims

Claims 1, 4, 5 and 7-10 are amended herein so as not to limit the scope of the claimed invention to a physical file system. The specification and drawings provide that in the mirror file system (MFS) of the present invention, the file systems linked by the MFS can be a local file system connected to a physical device, or a network file system exported by a remote system on a network. For example, Figure 2 illustrates that the mirror file system is mounted on top of a Unix File System (UFS) 18a and a Network File System (NFS) 18b to link these two file systems together so that the UFS and NFS contain identical data and are synchronized with each other in real time. Accordingly, the mirror file system of the present invention links any two or more file systems together and mirrors between them in real time.

Therefore, claims 1, 4, 5 and 7-10 are amended to broadly encompass any file system, including, but not limited to, a physical file system.

II. Rejections Under 35 U.S.C. §102

Claims 1-8 and 10-13 were rejected under 35 U.S.C. §102(b) as being anticipated by Provino et al. (U.S. Patent No. 5,778,384, hereinafter "Provino"). Applicant respectfully traverses this rejection for the following reasons.

An exemplary embodiment of the present invention discloses a virtual file system that links two or more file systems together and mirrors the file systems. For example, Figure 3 illustrates file system A 201 and file system B 202. File system A 201 includes a root directory /A containing two subdirectories b and i at a first level under the root directory /A. File system B 202 includes a root directory /X containing two subdirectories o and y at a first level under the root directory /X. Further

subdirectories and/or files are arranged at lower levels of the hierarchy in file systems A 201 and B 202. For instance, Figure 3 illustrates that subdirectory b of file system A 201 contains further subdirectories and/or files c-h, and subdirectory y of file system B 202 contains further subdirectories and/or files z and v. In the example of Figures 3 and 4, a portion of the structure of file system A 201 is to be linked to a portion of the structure of file system B 202. In particular, Figure 3 illustrates that structure B 220 of file system A 201 containing subdirectories and/or files b-h are to be linked with structure Y 221 of file system B 202 containing subdirectories and/or files y, z and v.

The disclosed embodiment provides that a mounting operation is performed for one of the file systems A 201 and system B 202. In particular, the disclosed embodiment provides that the directory y of file system B 202 is mounted onto the directory b of file system A 201, or the directory b of file system A 201 is mounted onto the directory y of file system B 202, as depicted in Figure 4. Consequently, as shown in Figure 4, the directory b of file system A now contains, in addition to subdirectories and/or files c-h, the subdirectories and/or files z and v originally contained in directory y of file system B 202, and the directory y of file system B 202 now contains, in addition to subdirectories and/or files z and v, subdirectories and/or files c-h originally contained in file system A 201.

As a result of this mounting operation, a virtual file system 203 is created. The virtual file system 203 has a data structure that contains all of the elements of the structure B 220 of the file system A 201 and the structure Y 221 of the file system B 202. Accordingly, the virtual file system 203 having a root directory of b/y links the structures B 220 and Y 221 together and mirrors the components contained in each of the structures B 220 and Y 221. The virtual file system 203 is thus created by mounting the components of each of the two file systems in a single directory, i.e., root directory b/y in the example of Figure 4. Therefore, the root directory b/y contains all of the mounted components within file systems A and B that are linked together, and the mounted components are mirrored in file systems A and B below the virtual file system 203 (see Figure 4).

Mirroring is achieved because a copy of each mounted component of the structures B 220 and Y 221 linked in the virtual file system 203 is stored in each of

the two file systems A 201 and B 202, as depicted in Figure 4. In particular, Figure 4 illustrates that copies of the components of directories and/or files b-h of structure B 220 are stored in file system B 202 below the virtual file system 203, and copies of the components of directories and/or files y, z and v are stored in file system A below the virtual file system 203. Accordingly, the virtual file system of the disclosed embodiment comprises a virtual file system data structure containing elements which respectively correspond to each of the mounted components.

Furthermore, each of the elements has an application interface data structure with two associated pointers that respectively point to application interface data structures of a corresponding component in each of the two file systems. In particular, the disclosed embodiment provides that every file or directory in the virtual file system 203 has a super *vnode* structure that is called an *mnode*. This *mnode* contains a *vnode* structure and two *vnode* pointers. The *vnode* structure comprises the *vnode* for the given file or directory within the virtual file system 203, and the two *vnode* pointers respectively point to the actual *vnode* of the corresponding file or directory within the two physical file systems A 201 and B 202. The two *vnode* pointers are represented in Figure 4 by the small black squares within the directories 231 to 233, 236 and 238, with arrows that point to their corresponding structures in the file systems A 201 and B 202.

In operation, when a request is made to access a file or directory under the b directory of file system A 201 or the y directory of file system B 202 within the virtual file system 203, the file system detects that these directories have the virtual file system 203 mounted on them. As a result, all file access requests are directed to the *vnode* for the virtual directory b/y of the virtual file system 203. Using the *mnode* data structure, the virtual file system 203 obtains the *vnodes* for both the directory b in file system A 201 and directory y in file system B 202. The requested operation, e.g. a write operation, is then sent to both of the *vnodes*. As a result, real time mirroring is effected between the corresponding pairs of elements, i.e. files or directories, in the two physical file systems A 201 and B 202.

The disclosed embodiment provides that the above-described mirror file system can be configured in a number of different ways, as depicted in Figures 5-10 of the application. Exemplary Figure 8 illustrates a client system 700 that uses a

mirror file system Z to access two physical file systems A and B, for example on different servers 710 and 720. In this configuration, file system A and file system B are exported to the client, where they are mounted by the virtual file system Z. In this configuration, operations performed by the client upon elements stored in the file system of either server are automatically mirrored to the file system of the other server, in a real-time fashion.

Independent claims 1, 4 and 11 recite various aspects of the above-described features.

Claim 1 recites a virtual file system which provides mirroring and linking of two file systems. The virtual file system of claim 1 comprises means for mounting components of each of the two file systems in a single directory.

The virtual file system of claim 1 also comprises a virtual file system data structure containing elements which respectively correspond to each of the mounted components, where each of the elements has an application interface data structure with two associated pointers that respectively point to application interface data structures of a corresponding component in each of the two file systems.

Provino discloses a virtual file system for accessing storage devices in a networked computer system. The virtual file system of Provino facilitates access of a virtual logic storage device that is identified by a virtual logic storage device identifier 41 and that has a virtual logic storage device file system including at least a portion of a remote file system maintained by another storage device in the computer network. In the system of Provino, computers 11(1) to 11(n) and 12 are connected by a communications link 13 to form a computer network 10 (see Figure 1). The system of Provino enables a computer 11 to access a file or file system maintained on a remote computer 11(1)-(n) and 12 by using the virtual logic storage device identifier 41 so that the computer 11 requesting access to the file or file system does not need to know the actual physical location of the computers 11(1)-(n) and 12 or a network protocol required to access the requested file or file system (see Column 6, lines 33-39).

The Examiner asserted that the virtual file system of claim 1 is disclosed in the unified directory structure of Figure 7 of Provino. Applicant respectfully asserts that this assertion is incorrect. The Examiner has mischaracterized the unified

directory structure of Figure 7 in rejecting claim 1. Accordingly, the formation of the unified directory structure of Provino is discussed first in order to illustrate the marked differences between the directory structure of Provino and the virtual file system of claim 1.

Each computer 11(1)-11(n) and 12 of Provino includes an application program 20, an operating system 21 and a network file system client module 23. The application program 20 issues an access request (ACC REQ) for a file, and the operating system 21 then determines whether the requested file is in a file system of a local storage device 22 or in a file system of a virtual storage device (see Column 6, lines 46-51). In other words, the operating system 21 determines whether the storage device to be accessed is a local storage device 22 within the computer or a virtual logical storage device within a remote computer in the network (see Column 6, line 64 to Column 7, line 1, and Figure 2). If the operating system 21 determines that the access request is for a virtual logical storage device, the operating system issues a remote access (REM ACC) to the network file system client module 23, which then determines whether the portion of file system of the virtual logical storage device containing the requested file has been mounted (see Column 7, lines 10-23). If the portion of the virtual logical storage device's file system is mounted, the network file system client module 23 initiates an access operation with the virtual logical storage device whose file system contains the requested file, in order to access the requested file (see Column 7, lines 23-35). If, on the other hand, the file system of the virtual logical storage device containing the requested file is not mounted, the network file system client module 23 issues an automount request (AUTOMOUNT REQ) to an automounter 25, which mounts a file system or needed portions thereof of a remote computer to the virtual logical storage device maintained by the computer 11(n) whose application program 20 issued the access request (see Column 7, lines 36-65).

The computer 11(n) whose application program 20 issued the access request stores an initialization file 40 including a virtual device identifier field 41 and a map file identifier field 44 (see Figure 3). The virtual device identifier field 41 includes device identifiers of virtual logical storage devices (i.e., device identifiers of remote computers), and the map file identifier field 44 includes a pointer to a file containing

an automount master map 50 (see Column 8, lines 30-34 and 56-62, and Figure 3). The automount master map 50 includes one or more entries 51(1) to 51(n) which are each associated with a high-level directory in the file system associated with a virtual logic storage device. Each entry 51(n) in the automount master map file 50 contains a number of fields, including a directory name field 52(n) containing a name of the topmost (root) directory in a directory tree, and an automount subdirectory name field 53(n) identifying an automount subdirectory map file 60 for subdirectories of the root directory identified in the directory name field 52(n) (see Column 8, line 67 to Column 9, line 16, and Figure 3). The automount subdirectory map file 60 includes an entry 61(1)-61(m) for each subdirectory of the root directory identified in the directory name field 52(n). Each entry 61(m) in the automount subdirectory map file 60 includes a subdirectory name field 62(m) identifying the name of a subdirectory under the root directory or an identifier of a remote computer containing the subdirectory, and a pointer field 63(m) containing a pointer to the remote file system 64 which is to be mounted for the subdirectory identified in the subdirectory name field 62(m) (see Column 9, lines 41-67).

Equipped with the initialization file 40, the automount master map file 50 and the automount subdirectory map file 60, the network file system client module 23 generates a directory tree 70 for the virtual logical storage devices. As shown in Figure 7, the directory tree 70 includes a series of entries 71(1) to 71(n). Each entry 71(n) in the directory tree 70 identifies the virtual logic storage device directory identified in the correspondingly-indexed root directory entry 51(n) of the automount master map file 50 (see Column 11, line 60 to Column 12, line 8). In other words, the directory DIR 1 of the directory tree 70 corresponds to the file system root directory identified in field 51(1) of the automount master map file 50. Each entry 71(n) of the directory tree 70 also includes "child" identifications corresponding to the subdirectories identified in the automount subdirectory map file 60 for the corresponding root directory 51(n) identified in the automount master map file 50 (see Column 12, lines 8-10, and field 71(3) in Figure 7).

In addition, each entry 71(n) of the directory tree 70 includes a "sibling" pointer to the entry 71(n+1) corresponding to the directory listed in the next entry 51(n+1) n the automount master map file 50. In other words, because the directory

DIR 1 of the directory tree 70 corresponds to the file system root directory identified in field 51(1) of the automount master file 50, the directory DIR 1 will contain a pointer to the root directory identified in field 51(2) of the automount master map file 50. In addition, each entry 71(n) also contains a subdirectory mount pointer containing information for mounting to the subdirectories of a remote file system, and the last entry of the directory tree 70 provides a mount pointer containing information for mounting the next remote file system, i.e., adding another file system to the directory tree 70 (see Column 12, lines 16-25).

Accordingly, the directory tree 70 in the virtual file system of Provino includes a plurality of successive fields 71(n) corresponding to each file system root directory identified in the automount master map file 50. The Examiner asserted that the directory tree 70 is a unified directory structure that results in the invention of claim 1. This assertion is incorrect for the following reasons.

First, the directory tree 70 of Provino results in mounting a local file system of a remote computer on a plurality of other directories within a single, unified (root) directory. In particular, Figure 7 illustrates that each entry 71(1)-71(3) corresponds to one file system as identified in the correspondingly-indexed root directory 51(n) contained in the automount master map file 50. Accordingly, entry 71(1) of the directory tree 70 would include an identification of a root directory of a first remote file system identified in entry 51(1), entry 71(2) of the directory tree 70 would include a root directory of a second remote file system identified in entry 51(2), entry 71(3) of the directory tree 70 would include a root directory of a third remote file system identified in entry 51(3), and so on, where each of the directories 71(1) to 71(3) could further include subdirectories of the respective entries 51(1) to 51(3). Therefore, the directory tree 70 of Provino provides one directory for one file system.

Second, the mounted file systems operate independently from one another, because there is no relationship between or among the mounted file systems other than being joined under the root directory 72, and because there is no file system mounted directly on the root directory 72.

Accordingly, Applicant respectfully submits that Provino clearly does not disclose or suggest means for mounting components of each of the two file systems in a single directory, as recited in claim 1. Instead, Provino mounts a first file system

51(1) in a first subdirectory 71(1) of the root directory 72, and then mounts a second file system 51(2) in a second subdirectory 71(2) of the root directory 72. In other words, Provino mounts two file systems 51(1) and 51(2) separately from each other in two different subdirectories 71(1) and 71(2) of the root directory 72 of the directory tree 70.

Third, the pointers contained in the directory tree 70 of Provino do not point to application interface data structures of a corresponding component in each of two file systems, as recited in claim 1. Instead, the pointers SIBLING and MOUNT PTR point to only one file system, not to the application interface data structures of a corresponding component in each of two file systems, as recited in claim 1. In particular, the SIBLING pointer merely points to the next file system 51(n+1) identified in the automount master map file 50. Similarly, the MOUNT PTR points to only one file system.

Furthermore, the CHILD pointers of Provino point to subdirectories within one file system, and therefore do not point to a component in each of two file systems.

Accordingly, Applicant respectfully submits that Provino also clearly does not disclose or suggest a virtual file system comprising a virtual file system data structure containing elements which respectively correspond to each of the mounted components, where each of the elements has an application interface data structure with two associated pointers that respectively point to application interface data structures of a corresponding component in each of the two file systems, as recited in claim 1.

Claim 4 recites a method for sharing files in a computer system. The method of claim 4 comprises the step of mounting components of each of two file systems in a single directory, such that a copy of each component is stored in each of the two file systems.

The method of claim 4 also comprises a step of receiving a request to perform a write operation on one of the components, and performing the write operation on both copies of the one component in the two file systems, respectively, in real time in response to the request.

As demonstrated above, Provino clearly does not disclose or suggest mounting components of each of two file systems in a single directory, as recited in claim 4.

Furthermore, as described above, each subdirectory 71(n) of the directory tree 70 contains one file system, and Provino does not disclose or suggest that a copy of each component in one file system 71(1) is stored in the other file system 71(2), and vice versa. This is a mirroring operation that is not disclosed, suggested or even contemplated by Provino.

As noted by the Examiner, Column 13, lines 40-48 describe a WRITE request. The Examiner, however, mischaracterizes the disclosure of WRITE operation of Provino in alleging that the write operation of claim 4 is disclosed in Provino. As described above, claim 4 recites performing the write operation on both copies of the one component in the two file systems, respectively, in real time in response to the request to perform a write operation on one of the components.

In contrast, Provino merely discloses that the network file system client module 23 generates and transfers a write message to a remote computer system 11(n') or server computer 12, and in response, the remote computer system 11(n') or server computer 12 stores the write request in its storage device (see Column 13, lines 40-48). Applicant respectfully submits that Provino cannot perform the write operation recited in claim 4. This is because Provino can write data to no more than one remote file system, since no directory can have more than one file system mounted on it.

Accordingly, in addition to not disclosing or suggesting the mounting step of claim 4, Provino also clearly does not disclose or suggest the performing step of claim 4.

Claim 11 recites a mirrored file system comprising a first server having a first local file system and a first physical storage device associated therewith, and a second server having a second local file system and a second physical storage device associated therewith.

The mirrored file system of claim 11 also comprises a client device having a virtual file system which mounts an imported file system from the first server and an

imported file system from the second server to provide a single point of access for components stored in each of the first and second local file systems.

Provino fails to disclose or suggest the client device of claim 11, because, as described above, a first local file system 51(1) imported from a first remote device is mounted in a first subdirectory 71(1) and a second local file system 51(2) imported from another remote device is mounted in a second subdirectory 71(2). The second subdirectory 71(2) is mapped onto the first subdirectory 71(1). As a result, the components stored in each of the first and second local file systems cannot be provided with a single point of access because they are not mounted in the same directory or subdirectory.

Accordingly, Provino clearly does not disclose or suggest a mirrored file system comprising a client device having a virtual file system which mounts an imported file system from the first server and an imported file system from the second server to provide a single point of access for components stored in each of the first and second local file systems, as recited in claim 11.

Therefore, for at least the foregoing reasons, Provino does not disclose or suggest each and every limitation of independent claims 1, 4 and 11.

Consequently, claims 1, 4 and 11 are clearly not anticipated by Provino, since Provino fails to disclose or suggest each and every limitation of claims 1, 4 and 11.

Furthermore, in view of the clear distinctions discussed above, Applicant respectfully submits that one skilled in the art would not have been motivated to modify Provino in such a manner as to result in, or otherwise render obvious, the inventions of claims 1, 4 and 11.

III. Rejections Under 35 U.S.C. § 103

Dependent claims 9 and 14-16 were rejected under 35 U.S.C. §103(a) as unpatentable over Provino. As demonstrated above, Provino clearly does not disclose or suggest each and every limitation of independent claims 1, 4 and 11. Consequently, Provino cannot disclose or suggest the limitations recited in claims 9 and 14-16 at least by virtue of their dependency from independent claims 4 and 11.

Therefore, for at least the foregoing reasons, Applicant respectfully submits that claims 1, 4 and 11, as well as claims 2, 3, 5-10 and 12-16 which depend therefrom, are clearly patentable over Provino.

IV. Conclusion

In view of the foregoing amendments and remarks, it is respectfully submitted that the present application is clearly in condition for allowance. An early notice thereof is respectfully solicited.

If, after reviewing this Amendment, the Examiner feels there are any issues remaining which must be resolved before the application can be passed to issue, the Examiner is respectfully requested to contact the undersigned by telephone in order to resolve such issues.

Respectfully submitted,

BUCHANAN INGERSOLL & ROONEY PC

Date: July 5, 2007

By: 

Jonathan R. Bowser
Registration No. 54574

P.O. Box 1404
Alexandria, VA 22313-1404
703 836 6620